Fine-Grained Graph Rationalization

Zhe Xu
zhexu3@illinois.edu
University of Illinois
Urbana-Champaign
Urbana, IL, USA

Menghai Pan
Yuzhong Chen
Huiyuan Chen
menpan@visa.com
yuzchen@visa.com
hchen@visa.com
Visa Research
Foster City, CA, USA

Yuchen Yan
yucheny5@illinois.edu
University of Illinois
Urbana-Champaign
Urbana, IL, USA

Mahashweta Das
mahdas@visa.com
Visa Research
Foster City, CA, USA

Hanghang Tong
htong@illinois.edu
University of Illinois
Urbana-Champaign
Urbana, IL, USA

Abstract

Rationale discovery is defined as finding a subset of the input data that maximally supports the prediction of downstream tasks. In the context of graph machine learning, *graph rationale* is defined as identifying the critical subgraph in the given graph topology. In contrast to the rationale subgraph, the remaining subgraph is named the *environment subgraph*. Graph rationalization can enhance the model performance because the mapping between the graph rationale and the prediction label is viewed as invariant, by definition. To ensure the discriminative power of the extracted rationale subgraphs, a key technique named *intervention* is applied, whose core idea is that given changing environment subgraphs, the semantics from the rationale subgraph is invariant, which guarantees the correct prediction result. However, most, if not all, of the existing graph rationalization methods develop their intervention strategies on the graph level, which is coarse-grained. In this paper, we propose FIne-grained Graph rationalization (FIG). Our idea is driven by the self-attention mechanism, which provides rich interactions between input nodes. Based on that, FIG can achieve node-level and virtual node-level intervention. Our experiments involve 7 real-world datasets, and the proposed FIG shows significant performance advantages compared to 13 baseline methods.

CCS Concepts

• **Mathematics of computing** → **Graph algorithms**; • **Computing methodologies** → **Neural networks**; • **Information systems** → **Data mining**.

Keywords

graph rationale discovery; graph neural network

CIKM '25, Seoul, Republic of Korea

ACM ISBN 979-8-4007-2040-6/2025/11
https://doi.org/10.1145/3746252.3761307

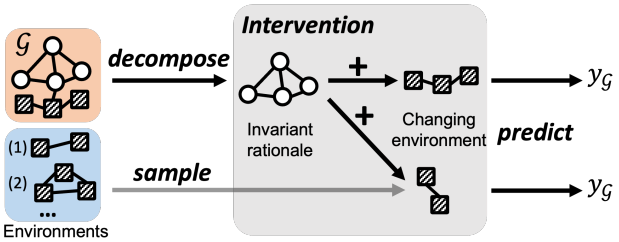


Figure 1: Illustration of the rationale/environment decomposition and intervention. Round nodes denote graph rationales, and square nodes (with stripes) denote the environments. The intervention aims to ensure the rationale from graph $\mathcal{G}$ truly has the discriminative power for the label $y_{\mathcal{G}}$.

ACM Reference Format:
Zhe Xu, Menghai Pan, Yuzhong Chen, Huiyuan Chen, Yuchen Yan, Mahashweta Das, and Hanghang Tong. 2025. Fine-Grained Graph Rationalization. In *Proceedings of the 34th ACM International Conference on Information and Knowledge Management (CIKM '25), November 10–14, 2025, Seoul, Republic of Korea.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3746252.3761307

1 Introduction

Rationale refers to a subset of the input features that play a crucial role in model predictions for downstream tasks [8, 18, 37, 49, 55, 77, 84]. In the context of graph machine learning, graph rationale is defined as a subgraph of the input graph containing the most task-relevant semantics. The application of graph rationale is broad; for example, it can greatly enhance model performance for graph-level tasks [33, 55, 61, 67, 81] by identifying the key components of the input graph. Additionally, the discovery of rationales can improve model explainability [37], as it highlights the parts of the input graph that significantly contribute to the final prediction.

Existing graph rationalization solutions [37, 55] employ a trainable *augmenter* to execute the rationale/environment decomposition. In this process, a node/edge mask is generated by the augmenter to decompose the given graph into a rationale graph and an environment graph. Inspired by the content-style decomposition [27], the key idea of graph rationalization is to preserve the

utility of the graph rationale when combined with changing environment graphs (see Figure 1). To achieve this, a technique named *intervention* is used, where the environment graph interacts with the rationale graph.

The intervention mechanism (named *intervener*) is essential in the graph rationalization process, as it must accurately represent the interaction between the rationale and the environment. Intuitively, **the intervener should work in an adversarial behavior** against the augmenter mentioned above, a point largely overlooked in the existing literature. If the intervener is more powerful, it can capture more detailed interactions between the rationale and environment subgraphs. Given such a powerful intervener, the augmenter is compelled to minimize these interactions between the graph rationale and the environment to obtain a "purer" graph rationale.

Unfortunately, existing works develop interveners in a coarse and non-parametric manner. After performing rationale/environment decomposition on the graph data, they compute graph-level embeddings for the rationale and environment subgraphs. The intervention is then designed as an interaction between these graph-level embeddings. For example, [37] adds the environment embedding into the rationale embedding as the intervened rationale embedding; [55] defines the intervened prediction as the Hadamard product between the predictions based on the rationale subgraph and the environment subgraph. We argue that **such a graph-level non-parametric intervention is insufficient** to fully represent the interaction between the rationale and environment graphs effectively.

In response to this limitation, we propose a fine-grained, parametric intervention named FIne-grained Graph rationalization (FIG). FIG draws inspiration from the self-attention module in the Transformer model, which captures interactions between input tokens. Building upon insights from Transformer [51] and its linear variant Linformer [54], FIG formulates the interaction between the rationale and environment subgraphs at the node-level or the virtual node-level. The two variants are named FIG-N and FIG-VN. Furthermore, to maximize the effectiveness of the intervention, we formulate a min-max game involving the node encoder, augmenter, intervener, and predictor, forcing the rationale subgraph to be as informative as possible.

We conduct comprehensive experiments on 7 graph-level benchmarks to evaluate the proposed approach and compare FIG-N/VN against 13 state-of-the-art baseline methods. The results demonstrate that FIG and its variants outperform the baseline methods, validating their superior performance. Our primary contributions in this paper are summarized as follows: (1) we address the graph rationalization problem via a fine-grained node/virtual node-level model, FIG; (2) a min-max game is proposed to boost the effectiveness of the proposed intervener; (3) extensive experiments covering 13 baseline methods and 7 real-world datasets demonstrate the efficacy of the proposed method.

2 Preliminaries

Section 2.1 introduces the notations used throughout this paper. Then, the classic Transformer architecture is introduced in Section 2.2, whose self-attention module is an important building block of our model. Last but not least, we introduce the overall ideas of existing graph rationalization works in Section 2.3.

2.1 Notations

We adopt the following notation conventions: bold uppercase letters for matrices and tensors (e.g., $\mathbf{A}$), bold lowercase letters for column vectors (e.g., $\mathbf{u}$), lowercase and uppercase letters in regular font for scalars (e.g., d, K), and calligraphic letters for sets (e.g., $\mathcal{T}$). To index vectors/matrices/tensors, we follow the syntax from NumPy (0-based). Specifically, $\mathbf{A}[p,:]$ and $\mathbf{A}[:,q]$ represent the p-th row and the q-th column of matrix $\mathbf{A}$ respectively; $\mathbf{A}[p,q]$ represents the entry at the p-th row and the q-th column. Similarly, $\mathbf{u}[p]$ denotes the p-th entry of vector $\mathbf{u}$. In addition, the slicing syntax for vectors/matrices/tensors is used. For example, for a matrix $\mathbf{A}$, $\mathbf{A}[i:j,:]$ denotes rows from the i-th row (included) to the j-th row (excluded) and $\mathbf{A}[:,:k]$ denotes all the columns before the k-th column. $\top$ denotes the transpose of matrices and vectors. $\odot$ represents the Hadamard product, and $\circ$ denotes function composition. We use $||$ for the concatenation operation, and the specific dimension of concatenation will be clarified based on the context.

An attributed graph can be represented as $\mathcal{G} = (\mathbf{A}, \mathbf{X}, \mathbf{E})$, where $\mathbf{A} \in \mathbb{R}^{n\times n}$ is the adjacency matrix, $\mathbf{X} \in \mathbb{R}^{n\times d_X}$ is the node feature matrix, and $\mathbf{E} \in \mathbb{R}^{n\times n\times d_E}$ is the edge feature tensor. Here, n denotes the number of nodes, and d_X (d_E) represents the dimensions of node (edge) features, respectively. This paper assumes the node and edge feature dimensions are the same (i.e., $d_X = d_E = d$) for brevity; if they differ, a fully connected layer can map them into a shared feature space. Our main focus in this paper is on graph property prediction tasks. The ground truth of a graph is represented by y.

2.2 Graph Transformer

The core modules of the Transformer architecture [51] are the self-attention layer and the feed-forward network layer. Given the input as a sequence of symbol representations $\mathbf{H} \in \mathbb{R}^{n\times d_H}$, it is first transformed into the query, key, and value matrices as

$$\mathbf{Q} = \mathbf{H}\mathbf{W}_Q, \quad \mathbf{K} = \mathbf{H}\mathbf{W}_K, \quad \mathbf{V} = \mathbf{H}\mathbf{W}_V, \tag{1}$$

where $\mathbf{W}_Q \in \mathbb{R}^{d_H\times d_Q}$, $\mathbf{W}_K \in \mathbb{R}^{d_H\times d_K}$, $\mathbf{W}_V \in \mathbb{R}^{d_H\times d_V}$. For brevity, we set $d_H = d_Q = d_K = d_V = d$. Then, the self-attention module is,

$$\mathbf{P} = \texttt{Attn}(\mathbf{H}) = \sigma\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d}}\right), \tag{2a}$$

$$\mathbf{H} \leftarrow \mathbf{P}\mathbf{V} + \mathbf{H}. \tag{2b}$$

Typically, the non-linearity σ is `Softmax`. The feed-forward network (FFN) updates the representations $\mathbf{H}$ as:

$$\mathbf{H} \leftarrow \texttt{FFN}(\mathbf{H}) + \mathbf{H}. \tag{3}$$

Optional techniques such as normalization [3, 24], dropout [47], and multi-head attention [51] are omitted here for brevity.

While Transformers were initially designed for sequence or set data with positional encoding, numerous techniques have since been introduced to adapt Transformers for graph data. Interested readers are referred to [41] for a detailed taxonomy. Interestingly, it is well-known in both the graph learning [9] and natural language processing communities [54, 78] that, from the message-passing perspective, the key idea of the Transformer architecture is to

construct a weighted complete graph, whose adjacency matrix is $\mathbf{P} = \sigma\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d}}\right)$ with complexity $O(n^2)$.

2.3 Invariant Rationale Discovery on Graphs

The graph rationale is a subgraph that encodes most downstream task-relevant semantics. A typical example is the functional groups in polymer graphs [37, 55], which fundamentally determines the chemical property of polymers. Mathematically, a given graph is decomposed into a rationale graph and an environment graph: $\mathcal{G} = \mathcal{G}_{\texttt{ra}} \cup \mathcal{G}_{\texttt{env}}$. Commonly, the graph embeddings on $\mathcal{G}_{\texttt{ra}}$ and $\mathcal{G}_{\texttt{env}}$ are computed as $\mathbf{h}_{\texttt{ra}}$ and $\mathbf{h}_{\texttt{env}}$. To ensure the rationale graph is invariant w.r.t. the prediction results when combined with different environments, a utility loss is minimized given the rationale embedding $\mathbf{h}_{\texttt{ra}}$ intervened by the environment embedding $\tilde{\mathbf{h}}_{\texttt{env}}$, i.e., $\min \mathcal{L}_{\texttt{util}}(\mathbf{h}_{\texttt{ra}} \xleftarrow{\text{intervene}} \tilde{\mathbf{h}}_{\texttt{env}})$. Here, $\tilde{\mathbf{h}}_{\texttt{env}}$ could be either from the same graph (i.e., $\tilde{\mathbf{h}}_{\texttt{env}} = \mathbf{h}_{\texttt{env}}$), or could be environment embeddings from other graphs, such as those in the batch.

A key difference among existing methods lies in the intervention operation, of which we mention two: (1) GREA [37] designs the intervention as sum, i.e., $\mathbf{h}_{\texttt{ra}}+\tilde{\mathbf{h}}_{\texttt{env}}$ and (2) DIR [55] designs an element-wise intervention: $\theta_{\texttt{pred}}(\mathbf{h}_{\texttt{ra}}) \odot \texttt{Sigmoid}(\theta_{\texttt{pred}}(\tilde{\mathbf{h}}_{\texttt{env}}))$, where $\odot$ is the Hadamard product and $\theta_{\texttt{pred}}$ is a predictor. The above intervention is graph-level because $\mathbf{h}_{\texttt{ra}}$ and $\tilde{\mathbf{f}}_{\texttt{env}}$ are **graph embeddings**. In Figure 2a, an overview of the GREA [37] is presented. As a comparison, we aim to design the intervention at finer grains (e.g., node level) to handle the interaction between rationale and environment graphs, which will be detailed as follows.

3 Proposed Model

This section introduces the proposed method, FIG. Figure 2b provides an overview of FIG, highlighting its four parametric modules: the encoder, augmenter, intervener, and predictor.

Encoder. The encoder, denoted as $\theta_{\texttt{enc}} : \mathcal{G} \to \mathbb{R}^{n\times d}$, accepts a graph data as input and outputs a node embedding matrix. There are various graph encoders available, such as graph neural networks (GNNs) [57] and graph Transformers [41]. From the methodology perspective, the encoder module is not the main contribution of this paper, so in this section, we do not specify a specific graph encoder $\theta_{\texttt{enc}}$.

Predictor. The predictor, denoted as $\theta_{\texttt{pred}} : \mathbb{R}^d \to \mathbb{R}^c$ takes as input a graph embedding and outputs a prediction vector/scalar. For graph regression tasks, $c = 1$; for graph classification tasks, c is the number of classes. A typical predictor is a multi-layer perceptron (MLP) with appropriate activation functions. Details of the encoder and predictor in our implementation are presented in Section 4.

In subsequent subsections, we will elaborate on the augmenter and intervener, two essential modules. Their detailed designs derive two variants of the proposed FIG.

3.1 Node-Level Variant: FIG-N

Node-level augmenter. The augmenter is a critical module of the proposed FIG. For the node-level variant, termed FIG-N, the augmenter's primary function is decomposing the node set into two distinct subsets: *rationale nodes and environment nodes*. This decomposition is operated by parameterizing the node-level augmenter as a learnable node partitioner, denoted by $\theta_{\texttt{aug-N}}$,

$$\mathbf{m} = \texttt{Sigmoid}(\texttt{MLP}(\mathbf{H}, \theta_{\texttt{aug-N}})), \tag{4}$$

whose input is the node embedding matrix $\mathbf{H} \in \mathbb{R}^{n\times d}$, and its output is a partition vector $\mathbf{m} \in [0,1]^n$. MLP is a multi-layer perceptron. Each entry within $\mathbf{m}$, such as $\mathbf{m}[i]$, denotes the probability of the i-th node being categorized as a rationale node.

For the partition vector $\mathbf{m}$, its top-K entries are indexed as $\texttt{idx}_{\texttt{ra}} = \texttt{argtopK}(\mathbf{m})$ which is used to index the rationale nodes from the node embedding matrix $\mathbf{H}$; naturally, the remaining nodes are categorized as the environment nodes whose indices are $\texttt{idx}_{\texttt{env}} = \{1,\dots,n\}/\texttt{idx}_{\texttt{ra}}$. K is a hyper-parameter whose impact is studied in Section 4.5. Also, in our implementation, we use a soft argtopK operation to maintain differentiability, whose details are in Section B.

Using the indices mentioned above, rationale and environment embeddings, denoted as $\mathbf{H}_{\texttt{ra}}$ and $\mathbf{H}_{\texttt{env}}$, respectively, can be extracted from the node embedding matrix $\mathbf{H}$:

$$\mathbf{H}_{\texttt{ra}} = \mathbf{H}[\texttt{idx}_{\texttt{ra}}, :] \in \mathbb{R}^{K\times d}, \tag{5a}$$

$$\mathbf{H}_{\texttt{env}} = \mathbf{H}[\texttt{idx}_{\texttt{env}}, :] \in \mathbb{R}^{(n-K)\times d}, \tag{5b}$$

Node-level intervener. The design of the fine-grained intervener draws inspiration from the Transformer architecture [51]. Explicitly, the node-level intervener ϕ is presented as,

$$\mathbf{H}_{\texttt{inter}}, \mathbf{P} = \texttt{Transformer}(\mathbf{H}_{\texttt{ra}}||\mathbf{H}_{\texttt{env}}), \tag{6a}$$

$$\textit{where} \quad \mathbf{P} = \texttt{Attn}(\mathbf{H}_{\texttt{ra}}||\mathbf{H}_{\texttt{env}}). \tag{6b}$$

In this representation, the operator || concatenates along the first dimension of the matrices $\mathbf{H}_{\texttt{ra}}$ and $\mathbf{H}_{\texttt{env}}$. We dub the Eqs. (1)-(3) as Transformer and $\mathbf{P}$ is the intermediate attention matrix from the self-attention layer (Eq. (6b)). Here, the self-attention module models the interactions between the rational nodes $\mathbf{H}_{\texttt{ra}}$ and the environment nodes $\mathbf{H}_{\texttt{env}}$. ϕ includes all the parameters of the Attn (Eq. (6b)) and FFN (Eq. (3)) modules. In some contexts where the attention matrix $\mathbf{P}$ is not explicitly used as an output, input/output of the intervener ϕ can be presented as $\mathbf{H}_{\texttt{inter}} = \phi(\mathbf{H}_{\texttt{ra}}||\mathbf{H}_{\texttt{env}})$.

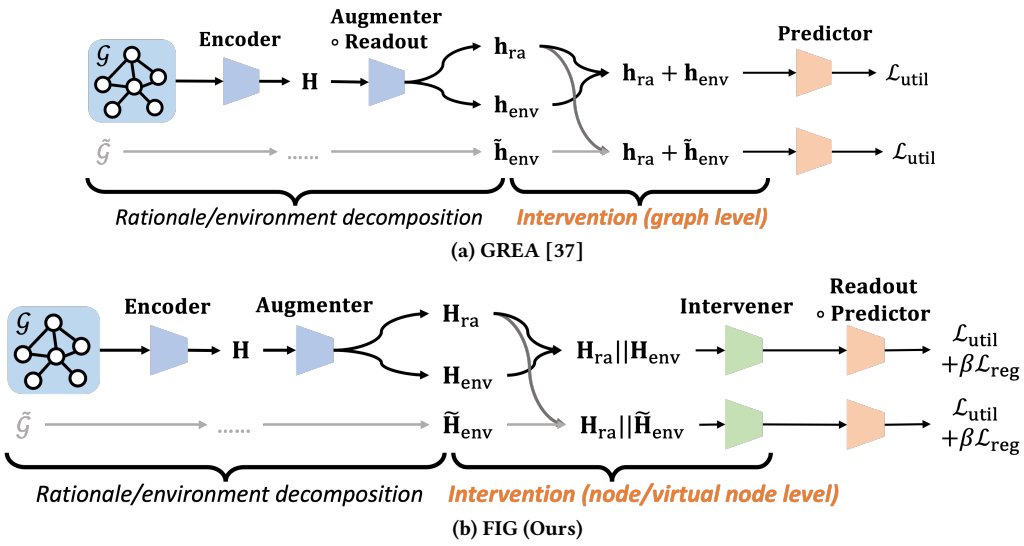
FIG-N optimization objective. The utility loss is computed as $\mathcal{L}_{\texttt{util}}(\mathbf{H}_{\texttt{ra}}||\mathbf{H}_{\texttt{env}}) = \mathcal{L}_{\texttt{task}}(\theta_{\texttt{pred}} \circ \texttt{Readout} \circ \phi(\mathbf{H}_{\texttt{ra}}||\mathbf{H}_{\texttt{env}})[:K], \mathbf{y})$, where $\mathcal{L}_{\texttt{task}}$ is the task-specific objective; e.g., it is the mean squared error for regression or the cross-entropy for classification. Notice that we select the top-K rows of the output of ϕ, i.e., the **intervened rationale part** and ideally, we expect $\phi(\mathbf{H}_{\texttt{ra}}||\mathbf{H}_{\texttt{env}})[:K] = \mathbf{H}_{\texttt{ra}}$.

As Figure 1 shows, invariant rationale discovery is to find the graph rationale so that the utility loss is minimized given **changing environments**. Thus, the utility objective is

$$\mathcal{L}_{\texttt{util}} = \mathcal{L}_{\texttt{util}}(\mathbf{H}_{\texttt{ra}}||\mathbf{H}_{\texttt{env}}) + \alpha\mathcal{L}_{\texttt{util}}(\mathbf{H}_{\texttt{ra}}||\tilde{\mathbf{H}}_{\texttt{env}}), \tag{7}$$

where $\tilde{\mathbf{H}}_{\texttt{env}}$ is the node embeddings from the changing environments. In practical implementations, $\tilde{\mathbf{H}}_{\texttt{env}}$ is the environment node embeddings from other graphs in the mini-batch. Additionally, to fully utilize the rich interactions from the fine-grained intervention module, we apply the following regularization term,

$$\mathcal{L}_{\texttt{reg}}(\mathbf{H}_{\texttt{ra}}||\mathbf{H}_{\texttt{env}}) = \mathbf{s}^\top\mathbf{P}(\mathbf{1}-\mathbf{s}) + (\mathbf{1}-\mathbf{s})^\top\mathbf{P}\mathbf{s}, \tag{8}$$



(a) GREA [37]

(b) FIG (Ours)

Figure 2: Pipeline comparison between existing work GREA and proposed FIG. ∘ denotes function composition. GREA designs the intervention at the graph level, and the proposed FIG designs the intervention at the node/virtual node level. The augmented environment $\tilde{\mathbf{H}}_{\mathsf{env}}$ is from another graph $\tilde{\mathcal{G}}$ (through the Encoder and Augmenter) in the batch.

where $\mathbf{P} \in \mathbb{R}^{n\times n}$ is the self-attention matrix from Eq. (6b),

$$\mathbf{s}[i] = \begin{cases} 1 & \text{if } i < K. \\ 0 & \text{otherwise.} \end{cases} \tag{9}$$

The binary $\mathbf{s}$ vector is used to designate **whether a particular row of the matrix $\mathbf{H}_{\mathsf{ra}}||\mathbf{H}_{\mathsf{env}}$ originates from the rationale nodes or the environment nodes**. The underlying notion of the regularization term Eq. (8) is to impose penalties on interactions between the rationale nodes and the environment nodes. Namely, these two terms $\mathbf{s}^\top\mathbf{P}(\mathbf{1}-\mathbf{s})$ and $(\mathbf{1}-\mathbf{s})^\top\mathbf{P}\mathbf{s}$ denote the **total weights on the links (i.e., cut) between the rationale and environment subgraphs**. To handle the changing environments, we introduce an additional regularization term on the changing environments as $\mathcal{L}_{\mathsf{reg}}(\mathbf{H}_{\mathsf{ra}}||\tilde{\mathbf{H}}_{\mathsf{env}})$ where $\tilde{\mathbf{H}}_{\mathsf{env}}$ is the environment node embeddings from another graph within the same mini-batch. Then, the total regularization term is

$$\mathcal{L}_{\mathsf{reg}} = \mathcal{L}_{\mathsf{reg}}(\mathbf{H}_{\mathsf{ra}}||\mathbf{H}_{\mathsf{env}}) + \mathcal{L}_{\mathsf{reg}}(\mathbf{H}_{\mathsf{ra}}||\tilde{\mathbf{H}}_{\mathsf{env}}), \tag{10}$$

and the final objective function is $\mathcal{L}_{\mathtt{util}} + \beta\mathcal{L}_{\mathsf{reg}}$. To fully harness the capabilities of the fine-grained parametric intervener, it is crucial to note—as highlighted in the introduction—that the behavior of the intervener ϕ operates in an **adversarial fashion** to the other modules. As a result, we formulate a min-max game that involves $\theta = \{\theta_{\mathsf{enc}}, \theta_{\mathsf{aug\text{-}N}}, \theta_{\mathsf{pred}}\}$ and ϕ as,

$$\min_{\theta}\max_{\phi} \quad \mathcal{L}_{\mathtt{util}} + \beta\mathcal{L}_{\mathsf{reg}}. \tag{11}$$

Here, the intervener ϕ is trained to **decrease the utility of the graph rationale by facilitating interactions between the rationale nodes and the environment nodes.** Conversely, the encoder, augmenter, and predictor (i.e., θ) are optimized in an opposing manner to the intervener's objectives.

Remark 1. The min-max objective is necessary. If it is replaced with the min objective, trivial solutions exist. E.g., if feature dimension d is large enough, ϕ could be an identity function with $\mathbf{P} = \mathbf{I}$, so that (1) $\mathcal{L}_{\mathsf{reg}} = 0$ and (2) we would always have $\phi(\mathbf{H}_{\mathsf{ra}}||\mathbf{H}_{\mathsf{env}})[:K] = \mathbf{H}_{\mathsf{ra}}$. However, in that case, the pipeline **degenerates to a regular graph classification/regression model, with no intervener.**

Algorithm 1: FIG-N single training step for graph $\mathcal{G}$

Input : a labelled graph $(\mathcal{G}, y)$, a sampled graph $\tilde{\mathcal{G}}$ from the same batch as $\mathcal{G}$, $\theta = \{\theta_{\mathsf{enc}}, \theta_{\mathsf{aug\text{-}N}}, \theta_{\mathsf{pred}}\}$, ϕ;
Output: updated θ and ϕ;
compute $\mathbf{H} = \theta_{\mathsf{enc}}(G)$ and $\tilde{\mathbf{H}} = \theta_{\mathsf{enc}}(\tilde{G})$;
compute $(\mathbf{H}_{\mathsf{ra}}, \mathbf{H}_{\mathsf{env}}) = \theta_{\mathsf{aug\text{-}N}}(\mathbf{H})$, $(\tilde{\mathbf{H}}_{\mathsf{ra}}, \tilde{\mathbf{H}}_{\mathsf{env}}) = \theta_{\mathsf{aug\text{-}N}}(\tilde{\mathbf{H}})$ via Eqs. (4), (5a), and (5b);
concatenate $\mathbf{H}_{\mathsf{ra}}||\mathbf{H}_{\mathsf{env}}$ and $\mathbf{H}_{\mathsf{ra}}||\tilde{\mathbf{H}}_{\mathsf{env}}$;
compute $\mathcal{L}_{\mathtt{util}}$ and $\mathcal{L}_{\mathsf{reg}}$ via Eqs. (7), (9), and (10);
update θ via gradient descent with $\frac{\partial(\mathcal{L}_{\mathtt{util}}+\beta\mathcal{L}_{\mathsf{reg}})}{\partial\theta}$;
update ϕ via gradient ascent with $\frac{\partial(\mathcal{L}_{\mathtt{util}}+\beta\mathcal{L}_{\mathsf{reg}})}{\partial\phi}$;

Complexity of FIG-N. As the encoder θ_{enc} and the predictor θ_{pred} are off-the-shelf, the FIG-N introduces two new modules: the node-level augmenter, $\theta_{\mathsf{aug\text{-}N}}$, and the Transformer-based intervener, ϕ. Notably, despite these additions, the increase in the number of parameters remains modest. The parameters for $\theta_{\mathsf{aug\text{-}N}}$ originate from the `MLP` defined in Eq. (4). In a configuration where the `MLP` has 3 layers with a feature dimension of d, the parameter count is $O(2d^2)$. The intervener ϕ, driven by the `Transformer` layer in Eq. (6a), has its parameters confined to $O(3d^2 + 2d^2) = O(5d^2)$, owing to its query, key, value projection matrices and the feed-forward net (`FFN` from Eq. (3), typically a 3-layered `MLP`).

A step-by-step algorithm for FIG-N is in Algorithm 1. In test phase, the output of $\theta_{\mathsf{pred}} \circ \mathtt{Readout} \circ \phi(\mathbf{H}_{\mathsf{ra}}||\mathbf{H}_{\mathsf{env}})$ is evaluated.

Algorithm 2: FIG-VN single training step for graph $\mathcal{G}$

Input : a labeled graph $(\mathcal{G}, y)$, a sampled graph $\tilde{\mathcal{G}}$ from the same batch as $\mathcal{G}$, $\theta = \{\theta_{\texttt{enc}}, \theta_{\texttt{aug-VN}}, \theta_{\texttt{pred}}\}$, ϕ;
Output : updated θ and ϕ;
compute $\mathbf{H} = \theta_{\texttt{enc}}(G)$ and $\tilde{\mathbf{H}} = \theta_{\texttt{enc}}(\tilde{G})$;
compute $(\mathbf{H}_{\texttt{ra}}, \mathbf{H}_{\texttt{env}}) = \theta_{\texttt{aug-VN}}(\mathbf{H})$, $(\tilde{\mathbf{H}}_{\texttt{ra}}, \tilde{\mathbf{H}}_{\texttt{env}}) = \theta_{\texttt{aug-VN}}(\tilde{\mathbf{H}})$ via Eqs. (12), (13), and (14);
concatenate $\mathbf{H}_{\texttt{ra}}||\mathbf{H}_{\texttt{env}}$ and $\mathbf{H}_{\texttt{ra}}||\tilde{\mathbf{H}}_{\texttt{env}}$;
compute $\mathcal{L}_{\texttt{util}}$ and $\mathcal{L}_{\texttt{reg}}$ via Eqs. (7), (9), and (10);
update θ via gradient descent with $\frac{\partial(\mathcal{L}_{\texttt{util}}+\beta\mathcal{L}_{\texttt{reg}})}{\partial\theta}$;
update ϕ via gradient ascent with $\frac{\partial(\mathcal{L}_{\texttt{util}}+\beta\mathcal{L}_{\texttt{reg}})}{\partial\phi}$;

3.2 Virtual Node-Level Variant: FIG-VN

In the previously introduced FIG-N, its augmenter decomposes the nodes into rationale nodes and environment nodes via a trainable node partitioner $\theta_{\texttt{aug-N}}$ so that the interaction is conducted at the node level (Eqs. (6a)) whose **dense attention matrix's complexity is quadratic in terms of the node numbers**. This section extends this idea to extract the graph rationale at the virtual node level, which has a **lower computation complexity (linear in terms of the node numbers)** compared to FIG-N and provides an intermediate intervention granularity between the node-level model (FIG-N) and the graph-level model (GREA [37]).

Virtual node-level augmenter. Our idea is partly inspired by the speedup technique from Linformer [54], which reformulates both the attention matrix and node (token) embedding matrix to dimensions of $\mathbb{R}^{n\times r}$ and $\mathbb{R}^{r\times d}$, respectively. This reformulation ensures that their multiplication scales linearly with the number of nodes (tokens) n because $r \ll n$ and d represents the feature dimension.

Building upon this insight, given node embeddings $\mathbf{H}$ from the encoder, the virtual node embeddings are:

$$\mathbf{H}_{\texttt{VN}} = \texttt{Softmax}(\mathbf{W}_{\texttt{N-VN}})\mathbf{H}. \tag{12}$$

Here, the row-wise applied $\texttt{Softmax}$ function, along with $\texttt{Softmax}(\mathbf{W}_{\texttt{N-VN}}) \in \mathbb{R}^{r\times n}$, yields a trainable matrix assigning n nodes to r virtual nodes, where r acts as a tunable hyper-parameter. More specifically, $\texttt{Softmax}(\mathbf{W}_{\texttt{N-VN}})$ can be viewed as **the adjacency matrix between r virtual nodes and n original nodes**; the virtual node embedding, $\mathbf{H}_{\texttt{VN}}$, is **the aggregation of original node embeddings after 1-step message passing**. In experiments, we set $r = 8$. As all the virtual node embeddings are learned, a subset of the r virtual nodes can be designated as rationale virtual nodes, whose rationality is data-driven by the intervention procedure discussed in subsequent subsections. For brevity, the initial K virtual nodes are deemed as rationale virtual nodes, while the last $r - K$ nodes are considered the environment virtual nodes; their embeddings are:

$$\mathbf{H}_{\texttt{ra}} = \mathbf{H}_{\texttt{VN}}[:K, :] \in \mathbb{R}^{K\times d}, \tag{13}$$

$$\mathbf{H}_{\texttt{env}} = \mathbf{H}_{\texttt{VN}}[K:, :] \in \mathbb{R}^{(r-K)\times d}. \tag{14}$$

Like the FIG-N, here K is a hyperparameter whose impact is studied in Section 4.5. The parameter of $\theta_{\texttt{aug-VN}}$ is only $\mathbf{W}_{\texttt{N-VN}}$.

Virtual node-level intervener. This section discusses the design of a virtual node-level intervener, similar to the intervener introduced in Section 3.1. The main difference is that the intervention here functions on the virtual nodes rather than the given real ones. We recall the rationale and environment virtual node embeddings $\mathbf{H}_{\texttt{ra}} \in \mathbb{R}^{K\times d}$ and $\mathbf{H}_{\texttt{env}} \in \mathbb{R}^{(r-K)\times d}$. Thanks to the property of the Transformer that it can **process sets with variable size**, the design of the virtual node-level intervener ϕ is similar to the node-level intervener as $\mathbf{H}_{\texttt{inter}}, \mathbf{P} = \texttt{Transformer}(\mathbf{H}_{\texttt{ra}}||\mathbf{H}_{\texttt{env}})$ or short as $\mathbf{H}_{\texttt{inter}} = \phi(\mathbf{H}_{\texttt{ra}}||\mathbf{H}_{\texttt{env}})$ if the attention matrix $\mathbf{P}$ is not used. Notably, for FIG-VN, $\mathbf{P} \in \mathbb{R}^{r\times r}$ describes the interaction among the r virtual nodes, whose complexity is only $O(r^2)$.

FIG-VN optimization objective. The output of $\theta_{\texttt{pred}} \circ \texttt{Readout} \circ \phi(\cdot)$ is used for minimizing the utility loss $\mathcal{L}_{\texttt{util}} = \mathcal{L}_{\texttt{util}}(\mathbf{H}_{\texttt{ra}}||\mathbf{H}_{\texttt{env}}) + \alpha\mathcal{L}_{\texttt{util}}(\mathbf{H}_{\texttt{ra}}||\tilde{\mathbf{H}}_{\texttt{env}})$, where $\mathcal{L}_{\texttt{util}}(\mathbf{H}_{\texttt{ra}}||\mathbf{H}_{\texttt{env}}) = \mathcal{L}_{\texttt{task}}(\theta_{\texttt{pred}} \circ \texttt{Readout} \circ \phi(\mathbf{H}_{\texttt{ra}}||\mathbf{H}_{\texttt{env}})[:K], \mathbf{y})$ and $\mathcal{L}_{\texttt{util}}(\mathbf{H}_{\texttt{ra}}||\tilde{\mathbf{H}}_{\texttt{env}})$ is defined similarly. For modeling the changing environment, $\tilde{\mathbf{H}}_{\texttt{env}}$ is the virtual node embeddings from other graphs in the mini-batch. Additionally, the previously proposed regularization term Eq, (8) can be extended to the virtual node-level variant: $\mathcal{L}_{\texttt{reg}}(\mathbf{H}_{\texttt{ra}}||\mathbf{H}_{\texttt{env}}) = \mathbf{s}^\top\mathbf{P}(\mathbf{1}-\mathbf{s}) + (\mathbf{1}-\mathbf{s})^\top\mathbf{P}\mathbf{s}$. The total regularization term, considering the changing environment $\tilde{\mathbf{H}}_{\texttt{env}}$, is $\mathcal{L}_{\texttt{reg}} = \mathcal{L}_{\texttt{reg}}(\mathbf{H}_{\texttt{ra}}||\mathbf{H}_{\texttt{env}}) + \mathcal{L}_{\texttt{reg}}(\mathbf{H}_{\texttt{ra}}||\tilde{\mathbf{H}}_{\texttt{env}})$. As the $\mathbf{P}$ depicts interactions among virtual nodes, we construct the rationale/environment indicator vector $\mathbf{s}$ analogously to Eq. (9). Put everything together, and the optimization objective of FIG-VN is $\min_\theta \max_\phi \ \mathcal{L}_{\texttt{util}} + \beta\mathcal{L}_{\texttt{reg}}$, where $\theta = \{\theta_{\texttt{enc}}, \theta_{\texttt{aug-VN}}, \theta_{\texttt{pred}}\}$.

Complexity of FIG-VN. As we mentioned the encoder $\theta_{\texttt{enc}}$ and the predictor $\theta_{\texttt{pred}}$ are off-the-shelf. Thus, the extra modules introduced by the FIG-VN are the virtual node-level augmenter $\theta_{\texttt{aug-VN}}$ and the Transformer-based intervener ϕ. The parameters for $\theta_{\texttt{aug-VN}}$ originate from the matrix $\mathbf{W}_{\texttt{N-VN}}$, as defined in Eq. (12), whose complexity is $O(nr)$, where n denotes the number of nodes. For practical implementation purposes, n is pre-set; it is set to 10× the average size of graphs from the dataset, and we truncate the input graphs if their sizes are larger than 10× the average size. The intervener ϕ parameters originate from the $\texttt{Transformer}$ layer, outlined in Eq. (6a). The number of parameters here is $O(5d^2)$, owing to its query, key, value projection matrices, and the feed-forward net (Eq. (3), typically a 3-layered $\texttt{MLP}$).

A step-by-step algorithm for FIG-VN is in Algorithm 2. In test phase, the output of $\theta_{\texttt{pred}} \circ \texttt{Readout} \circ \phi(\mathbf{H}_{\texttt{ra}}||\mathbf{H}_{\texttt{env}})$ is evaluated.

4 Experiments

In this section, the datasets and baseline methods are detailed first. Subsequent subsections evaluate the effectiveness of FIG, supplemented by efficiency studies, ablation studies, sensitivity studies, training convergence, and a visualization of the attention matrix. The detected rationales are visualized in the Appendix, Section 4.6.

4.1 Setup

In this paper, 7 publicly-accessible real-world datasets are used: (1) graph classification datasets molhiv [23], moltox21 [23], molbace [23], molbbbp [23] and (2) graph regression datasets ZINC [14], AQSOL [14], and mollipo [23]. We strictly follow the metrics and

Table 1: Effectiveness comparison (mean±std) with baseline methods. (↓) denotes the lower the better and (↑) denotes the higher the better. Statistics in grey are reported in the original papers. The best is bold, and the second best is underlined. N/A means the method cannot work on regression tasks.

	Graph Regression			Graph Classification			
Dataset	**ZINC**	**AQSOL**	**mollipo**	**molhiv**	**moltox21**	**molbace**	**molbbbp**
Metric	**MAE(↓)**	**MAE(↓)**	**RMSE(↓)**	**AUC(↑)**	**AUC(↑)**	**AUC(↑)**	**AUC(↑)**
GIN	0.350±0.008	1.237±0.011	0.783±0.017	77.1±1.5	75.6±0.9	80.7±1.2	69.5±1.0
GAT	0.723±0.010	1.638±0.048	0.923±0.011	75.0±0.5	72.2±0.6	75.3±0.8	67.1±0.6
GATv2	0.729±0.015	1.722±0.022	0.943±0.021	72.2±0.5	73.6±0.2	76.8±1.6	65.7±0.7
GatedGCN	0.579±0.023	1.533±0.035	0.819±0.033	74.8±1.6	75.0±0.8	81.2±1.2	68.3±0.9
GT	0.226±0.014	1.319±0.026	0.882±0.020	73.5±0.4	75.0±0.6	77.1±2.3	65.0±1.1
GraphiT	0.202±0.011	1.162±0.005	0.846±0.023	74.6±1.0	71.8±1.3	73.4±3.6	64.6±0.5
SAN	0.139±0.006	1.199±0.218	0.816±0.112	77.9±0.2	71.3±0.8	79.0±3.1	63.8±0.9
SAT	0.094±0.008	1.236±0.023	0.835±0.008	78.8±0.6	75.6±0.7	83.6±2.1	69.6±1.3
Graphormer	0.122±0.006	1.265±0.025	0.911±0.015	79.3±0.4	77.3±0.8	79.3±3.0	67.7±0.9
GraphTrans	0.192±0.011	1.233±0.052	0.915±0.032	78.1±0.5	76.4±0.8	78.0±1.8	70.5±0.9
GPS	**0.070±0.004**	1.032±0.007	0.780±0.021	78.8±1.0	75.7±0.4	79.6±1.4	69.6±1.1
DIR	N/A	N/A	N/A	77.1±0.6	73.1±0.2	74.8±0.3	70.5±1.4
GREA	0.227±0.020	1.177±0.019	0.769±0.025	79.3±0.9	78.2±0.9	82.4±2.4	69.9±1.8
FIG-N	0.095±0.008	**0.990±0.012**	0.708±0.013	80.1±0.7	**78.8±0.5**	**85.3±2.0**	**73.8±0.7**
FIG-VN	0.086±0.012	1.011±0.009	**0.706±0.009**	**80.2±1.0**	78.2±0.6	84.5±1.3	73.1±0.8

Table 2: Comparison of parameter count and FLOPs between the proposed augmenter/intervener and graph encoders.

Model	# Parameters	FLOPs
GIN	1, 708, 807	53, 008, 220
SAT	2, 790, 739	101, 520, 116
GraphTrans	2, 793, 307	111, 548, 906
GPS	3, 236, 239	133, 229, 235
$\{\theta_{\text{aug-N}}, \phi\}$	453, 001	31, 303, 800
$\{\theta_{\text{aug-VN}}, \phi\}$	363, 320	14, 558, 400

dataset split recommended by the given benchmarks. To be concrete, the area under the ROC curve (AUC) is the metric for datasets molhiv, moltox21, molbace, molbbbp; root-mean-square deviation (RMSE) is the metric for dataset mollipo; mean absolute error (MAE) is the metric for datasets ZINC and AQSOL. The detailed statistics of the datasets are given in Table 5 (Appendix). We report the average result with the standard deviation in 10 runs[1].

Our baseline methods include (1) 4 graph neural network models: GIN [59], GAT [52], GATv2 [7], and GatedGCN [6] (2) 7 graph Transformers: GT [13], GraphiT [40], SAN [29], SAT [9], Graphormer [74], GraphTrans [56], GPS [46], and (3) 2 graph rationale discovery methods: DIR [55] and GREA [37].

4.2 Effectiveness Study

The effectiveness comparison between the proposed FIG-N, FIG-VN, and baseline methods is provided in Table 1. To ensure a fair comparison, some pre-trained models, such as the pre-trained Graphormer [74], are excluded. As DIR [55] is designed to conduct interventions on the label prediction vectors, it cannot be directly applied to graph regression tasks.

We have several observations. First, our proposed FIG-N and FIG-VN consistently **outperform, or are at least on par with**, all the baseline methods on the graph classification and regression datasets. Second, the virtual-node variant FIG-VN **does not lead to significant performance degradation** compared to the node-level variant FIG-N.

4.3 Efficiency Study

A detailed comparison regarding the number of parameters and the FLOPs (floating point operations) is presented in Table 2, where we list 4 typical 5-layered encoders (GIN, SAT, GraphTrans, and GPS), and our proposed node-/virtual node-level augmenter, intervener modules (i.e., $\{\theta_{\text{aug-N}}, \phi\}$ and $\{\theta_{\text{aug-VN}}, \phi\}$). The comparison shows that our proposed interveners are **lightweight and only increase very minor computational costs**.

We also compare the training efficiency (iterations/second) of FIG-N and FIG-VN in Table 3 working with different encoders (GIN and GPS). The batch size is set as 32. We note that the inclusion of our proposed augmenter and intervener, represented as $\{\theta_{\text{aug-N}}, \phi\}$ or $\{\theta_{\text{aug-VN}}, \phi\}$, introduces a slight reduction in training speed. That is because the proposed parametric augmenter and intervener **increase the data pipeline steps**, as presented in figure 2b, and enlarge the computational graph for auto-gradient tools, such as PyTorch. Fortunately, the parameter count of the parametric augmenter and intervener is low, ensuring that the overall training speed of the model is not dramatically affected.

[1]The code is available at https://github.com/pricexu/FIG

Table 3: Wall-clock speed comparison (iterations/second) of encoder–intervener combinations. Higher is better. Numbers in parentheses indicate the speed drop compared to the vanilla encoder.

Encoder	Intervener	mollipo	molbace	molbbbp
GIN	None	29.38	25.62	27.11
	FIG-N	23.05 (↓6.33)	21.23 (↓4.39)	21.61 (↓5.50)
	FIG-VN	23.35 (↓6.03)	21.46 (↓4.16)	22.32 (↓4.79)
GPS	None	24.29	20.51	22.30
	FIG-N	19.57 (↓4.72)	17.83 (↓2.68)	18.67 (↓3.63)
	FIG-VN	19.93 (↓4.36)	18.16 (↓2.35)	18.88 (↓3.42)

Table 4: Ablation study (mean±std) of the proposed model FIG. (↓) indicates lower is better; (↑) indicates higher is better.

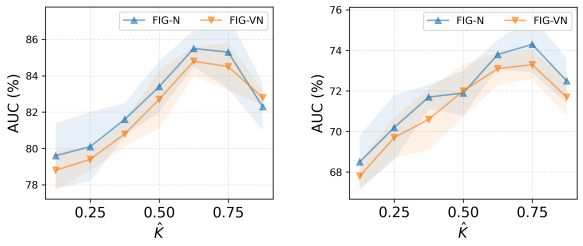
Dataset Metric	mollipo RMSE(↓)	molbace AUC(↑)	molbbbp AUC(↑)
$\theta_{\text{enc}} \circ \theta_{\text{pred}}$	0.780±0.021	79.6±1.4	70.5±0.9
FIG-N w/o reg	0.736±0.022	84.3±0.7	72.3±1.0
FIG-VN w/o reg	0.758±0.018	83.2±1.5	71.9±0.7
FIG-N w/ reg	0.708±0.013	85.3±2.0	73.8±0.7
FIG-VN w/ reg	0.706±0.009	84.5±1.3	73.1±0.8

4.4 Ablation Study

We conducted an ablation study on the proposed models, FIG-N and FIG-VN. We designed two ablated variants as baselines: (1) $\theta_{\text{enc}} \circ \theta_{\text{pred}}$ which is a pure composition of the encoder θ_{enc} and the predictor θ_{pred} *without any rationale discovery module*. Many of the existing graph classifiers are in this form, and here we select the GraphGPS [46], which also serves as the backbone of our FIG model. (2) FIG-N w/o reg and FIG-VN w/o reg which remove the regularization term (Eq. (10)) from the objective function. Our results in Table 4 highlight that (1) equipped with the proposed Transformer-based intervener, the model's performance improves across all the datasets; e.g., the AUC is improved from 79.6% to 84.3% (FIG-N) on the molbace dataset. (2) With the proposed regularization term, the model's performance can be improved further; e.g., the AUC of the FIG-N is improved from 72.3% to 73.8% on the molbbbp dataset.

4.5 Sensitivity Study

In this section, we carefully study the impact of hyperparameter K (from Eq. (5a), (5b), (13), and (14)), which determines the ratio of the rationale and environment subgraphs. In our implementation, we set $K = \texttt{round}(\hat{K} \times n)$ (for FIG-N) or $K = \texttt{round}(\hat{K} \times r)$ (for FIG-VN). We evaluate the model performance across varying $\hat{K}$ on the molbace and molbbbp datasets in Figure 3. We note that the model performance degrades if most nodes/virtual nodes are marked as the environment component. Similar performance degradation is observed if too many nodes/virtual nodes are marked as the rationale nodes (e.g., $\hat{K} = 0.875$). That is because for a small $\hat{K}$ (e.g., 0.25), too few nodes/virtual nodes are involved for the downstream tasks; for a large $\hat{K}$ (e.g., 1), the model degenerates to a vanilla graph encoder. The best performance is observed when $\hat{K}$ is 0.75 or 0.675.



(a) molbace **(b) molbbbp**

Figure 3: Performance of FIG-N/VN with different $\hat{K}$.

4.6 Training Convergence

We study the impact of the min-max objective on the training stability of FIG-N and FIG-VN. We monitor the training losses of both FIG-N and FIG-VN across datasets (molbace and molbbbp) using two encoders (GIN and GPS). The results, presented in Figure 4, demonstrate that the training remains stable even when θ (representing the encoder, augmenter, and predictor) and ϕ (representing the intervener) engage in a min-max game.

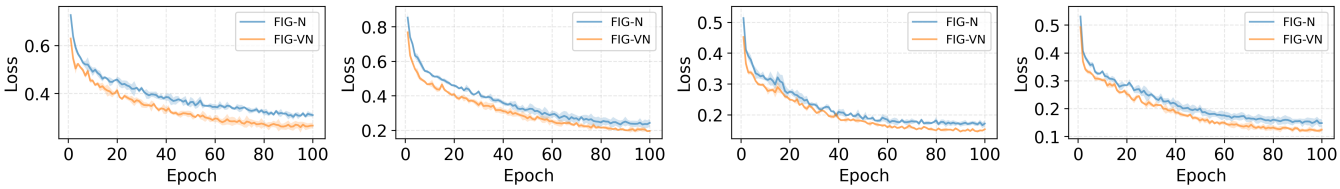
4.7 Attention Visualization

In this section, we aim to evaluate the significance of the regularization term by visualizing the attention matrix $\mathbf{P}$ of the intervener ϕ in Figure 5. For clarity in visualization, FIG-VN is chosen because its number of virtual nodes is predefined. Specifically, we set the number of virtual nodes r to 16 with K at 10, designating 10 virtual nodes to rationales and the remainder as environments. All the visualization results are obtained from the molbace dataset. It is worth noting that the attention matrix $\mathbf{P}$ is normalized row-wise by `Softmax`. From our observations, we highlight two insights:

- Interestingly, even in the absence of the regularization term, in Figure 5(a), interactions between rationales and environments appear significantly weaker than those within the rationales themselves. One potential explanation is the changing of the environment. In optimizing the utility loss $\mathcal{L}_{\texttt{util}} = \mathcal{L}_{\texttt{task}}(\mathbf{H}_{\texttt{ra}} || \mathbf{H}_{\texttt{env}}) + \alpha \mathcal{L}_{\texttt{task}}(\mathbf{H}_{\texttt{ra}} || \tilde{\mathbf{H}}_{\texttt{env}})$, the ever-changing environment ($\tilde{\mathbf{H}}_{\texttt{env}}$) might lead the model to minimize interactions between rationales and environments so that the utility of the rationale can be preserved.
- The first observation aligns with the motivation to introduce the regularization term, which aims to penalize rationale-environment interactions. When the proposed regularization term (Eq. (8)) is implemented, in Figure 5(b), there is **a noticeable decrease in rationale-environment interactions (the off-diagonal blocks in Figure 5)**. As our earlier ablation study demonstrated, this leads to improved model performance.

5 Related Work

This section introduces two topics related to this paper: (1) invariant learning on graphs and (2) Transformer on graphs.



(a) molbace, GIN (b) molbace, GPS (c) molbbbp, GIN (d) molbbbp, GPS

Figure 4: Training loss of FIG-N/VN with different datasets and encoders.

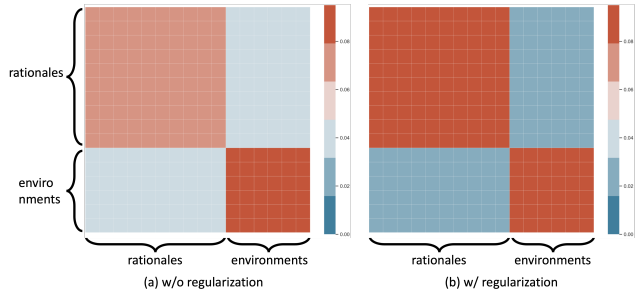


Figure 5: Heatmap of the adjacency matrix of the intervener ϕ. (a) without the regularization term Eq. (8) and (b) with the regularization term Eq. (8).

Invariant Learning on Graphs. Invariant learning, which has extensive applications in many fields such as computer vision [2, 8, 20, 27], is gaining more attention in the community of graph machine learning [4, 16, 34, 36, 39, 43, 50, 53, 58, 62–65, 68–72, 75, 76, 79, 80, 82, 83]. OOD-GNN [31] applies random Fourier features to eliminate the statistical dependence between relevant and irrelevant graph representations. [5] studies the graph invariant representation learning with a particular focus on the size discrepancies between the training/test graphs. DIR [55] decomposes the given graph into a rationale subgraph and an environment subgraph; after that, it uses a graph classifier to conduct prediction on the above two graphs respectively, and its intervention is via the Hadamard product between the prediction vectors. Similarly, GREA [37] conducts the rationale/environment decomposition at the node level, and its intervention operation is to directly add the environment graph embedding into the rationale graph embedding. In a similar vein, GIL [32] decomposes the given graph into the invariant and variant subgraphs; based on that, it infers the environment label, as input of the invariance regularization term [28]. Furthermore, invariant learning can also benefit graph contrastive learning [25, 26, 48, 85, 86] to enhance data augmentation. Using invariant learning to address the OOD generalization challenges on graph [12, 19, 21, 30, 49, 73, 77, 87] is promising, but our paper does not concentrate on this setting, and we leave it as future work.

Transformer on Graphs. Transformer [1, 11, 38, 44, 45, 51] has achieved significant success in various domains, including natural language processing [51], computer vision [22], and more [10, 15, 17, 35, 42, 60, 66]. In recent years, there has been a surge of interest in enhancing graph machine learning methods by leveraging the capabilities of Transformers. For example, GraphTrans [56] concatenates the Transformer architecture after various message-passing neural networks; GPS [46] proposes a powerful layer which operates both a graph convolution layer and a Transformer layer in parallel; GT [13] generalizes the attention module on graphs via concatenating the spectral vectors with the raw node features and computing the attention weights on all the existing edges; Graphormer [74] encodes both the node centrality and node-pair shortest path into the Transformer architecture. A comprehensive survey about the graph Transformer can be found in [41].

6 Conclusion

This paper studies the invariant rationale discovery problem on graphs. Distinct from existing methods, our solutions (FIG-N and FIG-VN) are designed at more fine-grained levels, specifically at the node and virtual node levels, so that they can better model the interactions between the rationale and environment subgraphs. More importantly, we formulate the intervener and the other model modules in a min-max game, which can significantly improve the quality of the extracted graph rationales. Comprehensive experiments on 7 real-world datasets illustrate the effectiveness of the proposed method compared to 13 baseline methods. Our evaluation of the proposed augmenters' and interveners' efficiency shows that they can largely retain the overall training efficiency.

Acknowledgments

This work is partially supported by NSF (2134079). The content of the information in this document does not necessarily reflect the position or the policy of the Government, and no official endorsement should be inferred. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation hereon.

A Hardware and Implementations.

We implement FIG-N, FIG-VN, and all the baseline methods in PyTorch and PyTorch-geometric. All the efficiency study results are from one NVIDIA Tesla V100 SXM2-32GB GPU on a server with 96 Intel(R) Xeon(R) Gold 6240R CPU @ 2.40GHz processors and 1.5T RAM. We directly use those statistics when baseline methods have pre-existing results for specific datasets. In cases where such results are absent, we implement the models based on either the available code or details described in the associated publications.

We first introduce some shared implementations among FIG-N/VN and baseline methods. The batch size is set as 32, and the weight decay is set as 0. The hidden dimension is set as 300. The dropout rate for both the encoder and the intervener is searched between $\{0, 0.5\}$. The pooling is searched between $\{\texttt{mean}, \texttt{sum}\}$. The normalization is searched between the batch normalization [24] and layer normalization [3]. The learning rate is initialized as 0.0001

Table 5: Statistics of datasets used in our experiments. We report the average number of nodes and edges per graph, input feature dimensions, class count (if applicable), and the train/validation/test splits.

Dataset	# Graphs	Avg. Nodes	Avg. Edges	Features	# Classes	Split (train/val/test)	Metric
ZINC	12,000	23.2	49.8	21 (node), 4 (edge)	N/A	10,000 / 1,000 / 1,000	MAE
AQSOL	9,833	17.6	35.8	65 (node), 5 (edge)	N/A	7,836 / 998 / 999	MAE
mollipo	4,200	27.0	59.0	9 (node), 3 (edge)	N/A	3,360 / 420 / 420	RMSE
molhiv	41,127	25.5	54.9	9 (node), 3 (edge)	2	32,901 / 4,113 / 4,113	AUC
moltox21	7,831	18.6	38.6	9 (node), 3 (edge)	2	6,264 / 783 / 784	AUC
molbace	1,513	34.4	73.7	9 (node), 3 (edge)	2	1,210 / 151 / 152	AUC
molbbbp	2,039	24.1	51.9	9 (node), 3 (edge)	2	1,631 / 204 / 204	AUC

and will decay by a factor $\frac{1}{4}$ if the validation performance is not improved in 10 epochs. Next, we detail the implementation of FIG-N/VN and baseline methods.

The statistics of the datasets used are presented in Table 5.

Implementation of FIG-N/VN. The encoder is set as GPS [46] on ZINC, AQSOL, mollipo, molhiv, molbace, and set as GraphTrans [56] on moltox21 and molbbbp. We follow the typical design for the predictor module as a 3-layered MLP with ReLU activation in the intermediate layers. In our implementation, we set $\beta = \frac{2\times\hat{\beta}}{n\times(n-1)}$ (FIG-N) or $\beta = \frac{2\times\hat{\beta}}{r\times(r-1)}$ (FIG-VN). The α and $\hat{\beta}$ are searched between $[0.2, 2]$, step size 0.2. In our implementation, the K is set as $K = \texttt{round}(\hat{K}\times n)$ (for FIG-N) or $K = \texttt{round}(\hat{K}\times r)$ (for FIG-VN). r is searched between $\{8, 16, 32\}$ for FIG-VN. We have a detailed sensitivity study to explore the best selection of $\hat{K}$ in Section 4.5, which shows the best $\hat{K}$ is around 0.75.

Implementation of baseline methods. We search the number of layers of GIN [59], GAT [52], GATv2 [7], GatedGCN [6], DIR [55], and GREA [37] between $\{2, 3, 5, 10\}$ and report the best performance, considering configurations both with and without a virtual node connecting to all the given nodes.

Regarding the Transformer-based baselines (GT [13], GraphiT [40], SAN [29], SAT [9], Graphormer [74], GraphTrans [56], GPS [46]), for the (absolute or relative) positional encoding, we adhere to the suggestions made in their original papers. We also searched the number of layers between $\{2, 3, 5, 10\}$.

Our GIN, GAT, GATv2, and GatedGCN implementations are from the PyTorch-geometric package. Our implementations of GT, GraphiT, SAN, SAT, Graphormer, GraphTrans, GPS, DIR, and GREA are adapted from publicly available code.

B Soft argtop-K Trick

In the main content, for the FIG-N model, the augmenter will generate the partition vector $\mathbf{m}$ which is then used to partition the node embedding matrix via selecting the top-K indices from $\mathbf{m}$:

$$\mathbf{m} = \texttt{Sigmoid}(\texttt{MLP}(\mathbf{H}, \theta_{\texttt{aug-N}})) \in \mathbb{R}^n \tag{15}$$

$$\texttt{idx}_{\texttt{ra}} = \texttt{argtopK}(\mathbf{m}) \in \mathbb{N}_+^K \tag{16}$$

$$\mathbf{H}_{\texttt{ra}} = \mathbf{H}[\texttt{idx}_{\texttt{ra}}, :] \in \mathbb{R}^{K\times d} \tag{17}$$

$$\mathbf{H}_{\texttt{env}} = \mathbf{H}[\texttt{idx}_{\texttt{env}}, :] \in \mathbb{R}^{(n-K)\times d} \tag{18}$$

The hard `argtopK` **breaks the differentiability of the model so that $\theta_{\texttt{aug-N}}$ has no gradient**. Here, we present a soft argtop-K trick which is inspired by the soft top-K trick[2]. Overall, the key ideas are as follows,

(1) The index vector $\mathbf{idx}_{\texttt{ra}} \in \mathbb{N}_+^K$ can be viewed as **a list of one-hot index encoding idx** $\in \{0, 1\}^{K\times n}$ whose every row is a one-hot vector. If the i-th row's j-th element is 1, it means the j-th element in $\mathbf{m}$ is the i-th largest element in $\mathbf{m}$. Then, we can use the matrix multiplication to index the matrix $\mathbf{H}$:

$$\mathbf{H}[\texttt{idx}_{\texttt{ra}}, :] \Longleftrightarrow \mathbf{idx}\cdot\mathbf{H} \in \mathbb{R}^{K\times d} \tag{19}$$

(2) We aim to find a soft and differentiable matrix to approximate **idx**. The trick is to use the fact that every row of **idx** is one-hot, which can be **approximated by the output of the softmax function**. Hence, the implementation is to call the softmax K times repeatedly.

(3) The parameterization trick can be used: y_hard - y_soft.detach() + y_soft, whose main idea is to ensure that (1) the forward process uses the hard indexing and (2) the backpropagation updates the soft indices.

We provide the exemplar code as follows.

```
import torch
def soft_arg_top_K(K, H, m):
    """
    Select the argtop K indices from the vector m, and index
          the node embedding matrix H.

    Args:
        K: number of nodes selected
        H: node embedding matrix (n, d)
        m: masking vector (n,)

    Returns:
        Aggregated embedding of shape (K, d).
    """
    top_K_indices = []
    for i in range(K):
        top1_index_soft = torch.nn.Softmax(m)
        top1_index_hard = torch.zeros_like(top1_index_soft)
        top1_index_hard[torch.argmax(top1_index_soft)] = 1
        top1_index = top1_index_hard - top1_index_soft.detach
             () + top1_index_soft
        top_K_indices.append(top1_index)
        m = m - top1_index_hard * 1e6
    top_K_indices = torch.stack(top_K_indices, dim=0)
    return torch.mm(top_K_indices, H)
```

[2]https://github.com/ZIB-IOL/merlin-arthur-classifiers/blob/main/soft-topk.py

C GenAI Usage Disclosure

We used ChatGPT and Grammarly to assist with grammar correction and word refinement. Microsoft Copilot was used to support code development. We take full responsibility for the accuracy and integrity of all content presented in this work.

References

[1] Mengting Ai, Tianxin Wei, Yifan Chen, Zhichen Zeng, Ritchie Zhao, Girish Varatkar, Bita Darvish Rouhani, Xianfeng Tang, Hanghang Tong, and Jingrui He. 2025. ResMoE: Space-efficient Compression of Mixture of Experts LLMs via Residual Restoration. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining, V.1, KDD 2025, Toronto, ON, Canada, August 3-7, 2025*, Yizhou Sun, Flavio Chierichetti, Hady W. Lauw, Claudia Perlich, Wee Hyong Tok, and Andrew Tomkins (Eds.). ACM, 1–12. doi:10.1145/3690624.3709196

[2] Martín Arjovsky, Léon Bottou, Ishaan Gulrajani, and David Lopez-Paz. 2019. Invariant Risk Minimization. *CoRR* abs/1907.02893 (2019). arXiv:1907.02893

[3] Lei Jimmy Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. 2016. Layer Normalization. *CoRR* abs/1607.06450 (2016). arXiv:1607.06450

[4] Wenxuan Bao, Zhichen Zeng, Zhining Liu, Hanghang Tong, and Jingrui He. 2025. Matcha: Mitigating Graph Structure Shifts with Test-Time Adaptation. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net. https://openreview.net/forum?id=EpgoFFUM2q

[5] Beatrice Bevilacqua, Yangze Zhou, and Bruno Ribeiro. 2021. Size-Invariant Graph Representations for Graph Classification Extrapolations. In *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event (Proceedings of Machine Learning Research, Vol. 139)*, Marina Meila and Tong Zhang (Eds.). PMLR, 837–851.

[6] Xavier Bresson and Thomas Laurent. 2017. Residual Gated Graph ConvNets. *CoRR* abs/1711.07553 (2017). arXiv:1711.07553

[7] Shaked Brody, Uri Alon, and Eran Yahav. 2022. How Attentive are Graph Attention Networks?. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net.

[8] Shiyu Chang, Yang Zhang, Mo Yu, and Tommi S. Jaakkola. 2020. Invariant Rationalization. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event (Proceedings of Machine Learning Research, Vol. 119)*. PMLR, 1448–1458.

[9] Dexiong Chen, Leslie O'Bray, and Karsten M. Borgwardt. 2022. Structure-Aware Transformer for Graph Representation Learning. In *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA (Proceedings of Machine Learning Research, Vol. 162)*, Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvári, Gang Niu, and Sivan Sabato (Eds.). PMLR, 3469–3489.

[10] Huiyuan Chen, Zhe Xu, Chin-Chia Michael Yeh, Vivian Lai, Yan Zheng, Minghua Xu, and Hanghang Tong. 2024. Masked Graph Transformer for Large-Scale Recommendation. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2024, Washington DC, USA, July 14-18, 2024*, Grace Hui Yang, Hongning Wang, Sam Han, Claudia Hauff, Guido Zuccon, and Yi Zhang (Eds.). ACM, 2502–2506. doi:10.1145/3626772.3657971

[11] Lingjie Chen, Ruizhong Qiu, Siyu Yuan, Zhining Liu, Tianxin Wei, Hyunsik Yoo, Zhichen Zeng, Deqing Yang, and Hanghang Tong. 2024. WAPITI: A Watermark for Finetuned Open-Source LLMs. *CoRR* abs/2410.06467 (2024). arXiv:2410.06467 doi:10.48550/ARXIV.2410.06467

[12] Kaize Ding, Zhe Xu, Hanghang Tong, and Huan Liu. 2022. Data Augmentation for Deep Graph Learning: A Survey. *SIGKDD Explor.* 24, 2 (2022), 61–77. doi:10.1145/3575637.3575646

[13] Vijay Prakash Dwivedi and Xavier Bresson. 2020. A Generalization of Transformer Networks to Graphs. *CoRR* abs/2012.09699 (2020). arXiv:2012.09699

[14] Vijay Prakash Dwivedi, Chaitanya K. Joshi, Anh Tuan Luu, Thomas Laurent, Yoshua Bengio, and Xavier Bresson. 2023. Benchmarking Graph Neural Networks. *J. Mach. Learn. Res.* 24 (2023), 43:1–43:48.

[15] Dongqi Fu, Liri Fang, Zihao Li, Hanghang Tong, Vetle I Torvik, and Jingrui He. 2024. What Do LLMs Need to Understand Graphs: A Survey of Parametric Representation of Graphs. *arXiv preprint arXiv:2410.12126* (2024).

[16] Dongqi Fu and Jingrui He. 2021. SDG: A Simplified and Dynamic Graph Neural Network. In *SIGIR '21: The 44th International ACM SIGIR Conference on Research and Development in Information Retrieval, Virtual Event, Canada, July 11-15, 2021*, Fernando Diaz, Chirag Shah, Torsten Suel, Pablo Castells, Rosie Jones, and Tetsuya Sakai (Eds.). ACM, 2273–2277. doi:10.1145/3404835.3463059

[17] Dongqi Fu, Zhigang Hua, Yan Xie, Jin Fang, Si Zhang, Kaan Sancak, Hao Wu, Andrey Malevich, Jingrui He, and Bo Long. 2024. VCR-Graphormer: A Mini-batch Graph Transformer via Virtual Connections. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net. https://openreview.net/forum?id=SUUrkC3STJ

[18] Dongqi Fu, Zhe Xu, Bo Li, Hanghang Tong, and Jingrui He. 2020. A View-Adversarial Framework for Multi-View Network Embedding. In *CIKM '20: The 29th ACM International Conference on Information and Knowledge Management, Virtual Event, Ireland, October 19-23, 2020*, Mathieu d'Aquin, Stefan Dietze, Claudia Hauff, Edward Curry, and Philippe Cudré-Mauroux (Eds.). ACM, 2025–2028. doi:10.1145/3340531.3412127

[19] Dongqi Fu, Zhe Xu, Hanghang Tong, and Jingrui He. 2023. Natural and Artificial Dynamics in GNNs: A Tutorial. In *Proceedings of the Sixteenth ACM International Conference on Web Search and Data Mining, WSDM 2023, Singapore, 27 February 2023 - 3 March 2023*, Tat-Seng Chua, Hady W. Lauw, Luo Si, Evimaria Terzi, and Panayiotis Tsaparas (Eds.). ACM, 1252–1255. doi:10.1145/3539597.3572726

[20] Yaroslav Ganin and Victor S. Lempitsky. 2015. Unsupervised Domain Adaptation by Backpropagation. In *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015 (JMLR Workshop and Conference Proceedings, Vol. 37)*, Francis R. Bach and David M. Blei (Eds.). JMLR.org, 1180–1189.

[21] Shurui Gui, Xiner Li, Limei Wang, and Shuiwang Ji. 2022. GOOD: A Graph Out-of-Distribution Benchmark. In *NeurIPS*.

[22] Kai Han, Yunhe Wang, Hanting Chen, Xinghao Chen, Jianyuan Guo, Zhenhua Liu, Yehui Tang, An Xiao, Chunjing Xu, Yixing Xu, Zhaohui Yang, Yiman Zhang, and Dacheng Tao. 2023. A Survey on Vision Transformer. *IEEE Trans. Pattern Anal. Mach. Intell.* 45, 1 (2023), 87–110. doi:10.1109/TPAMI.2022.3152247

[23] Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. 2020. Open Graph Benchmark: Datasets for Machine Learning on Graphs. *CoRR* abs/2005.00687 (2020). arXiv:2005.00687

[24] Sergey Ioffe and Christian Szegedy. 2015. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015 (JMLR Workshop and Conference Proceedings, Vol. 37)*, Francis R. Bach and David M. Blei (Eds.). JMLR.org, 448–456.

[25] Baoyu Jing, Yuchen Yan, Kaize Ding, Chanyoung Park, Yada Zhu, Huan Liu, and Hanghang Tong. 2024. Sterling: Synergistic Representation Learning on Bipartite Graphs. In *Thirty-Eighth AAAI Conference on Artificial Intelligence, AAAI 2024, Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence, IAAI 2024, Fourteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2014, February 20-27, 2024, Vancouver, Canada*, Michael J. Wooldridge, Jennifer G. Dy, and Sriraam Natarajan (Eds.). AAAI Press, 12976–12984. doi:10.1609/AAAI.V38I12.29195

[26] Baoyu Jing, Yuchen Yan, Yada Zhu, and Hanghang Tong. 2022. COIN: Co-Cluster Infomax for Bipartite Graphs. *CoRR* abs/2206.00006 (2022). arXiv:2206.00006 doi:10.48550/ARXIV.2206.00006

[27] Jean Kaddour, Aengus Lynch, Qi Liu, Matt J. Kusner, and Ricardo Silva. 2022. Causal Machine Learning: A Survey and Open Problems. *CoRR* abs/2206.15475 (2022). arXiv:2206.15475 doi:10.48550/arXiv.2206.15475

[28] Masanori Koyama and Shoichiro Yamaguchi. 2020. Out-of-Distribution Generalization with Maximal Invariant Predictor. *CoRR* abs/2008.01883 (2020). arXiv:2008.01883

[29] Devin Kreuzer, Dominique Beaini, William L. Hamilton, Vincent Létourneau, and Prudencio Tossou. 2021. Rethinking Graph Transformers with Spectral Attention. In *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan (Eds.). 21618–21629.

[30] Haoyang Li, Xin Wang, Ziwei Zhang, and Wenwu Zhu. 2022. Out-Of-Distribution Generalization on Graphs: A Survey. *CoRR* abs/2202.07987 (2022). arXiv:2202.07987

[31] Haoyang Li, Xin Wang, Ziwei Zhang, and Wenwu Zhu. 2023. OOD-GNN: Out-of-Distribution Generalized Graph Neural Network. *IEEE Trans. Knowl. Data Eng.* 35, 7 (2023), 7328–7340. doi:10.1109/TKDE.2022.3193725

[32] Haoyang Li, Ziwei Zhang, Xin Wang, and Wenwu Zhu. 2022. Learning Invariant Graph Representations for Out-of-Distribution Generalization. In *NeurIPS*.

[33] Ting-Wei Li, Ruizhong Qiu, and Hanghang Tong. 2025. Model-Free Graph Data Selection under Distribution Shift. *CoRR* abs/2505.17293 (2025). arXiv:2505.17293 doi:10.48550/ARXIV.2505.17293

[34] Zihao Li, Dongqi Fu, Mengting Ai, and Jingrui He. 2025. APEX^2: Adaptive and Extreme Summarization for Personalized Knowledge Graphs. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining, V.1, KDD 2025, Toronto, ON, Canada, August 3-7, 2025*, Yizhou Sun, Flavio Chierichetti, Hady W. Lauw, Claudia Perlich, Wee Hyong Tok, and Andrew Tomkins (Eds.). ACM, 741–752. doi:10.1145/3690624.3709213

[35] Zihao Li, Xiao Lin, Zhining Liu, Jiaru Zou, Ziwei Wu, Lecheng Zheng, Dongqi Fu, Yada Zhu, Hendrik F. Hamann, Hanghang Tong, and Jingrui He. 2025. Language in the Flow of Time: Time-Series-Paired Texts Weaved into a Unified Temporal Narrative. *CoRR* abs/2502.08942 (2025). arXiv:2502.08942 doi:10.48550/ARXIV.2502.08942

[36] Zihao Li, Lecheng Zheng, Bowen Jin, Dongqi Fu, Baoyu Jing, Yikun Ban, Jingrui He, and Jiawei Han. 2025. Can Graph Neural Networks Learn Language with

Extremely Weak Text Supervision?. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2025, Vienna, Austria, July 27 - August 1, 2025*, Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (Eds.). Association for Computational Linguistics, 11138–11165. https://aclanthology.org/2025.acl-long.545/

[37] Gang Liu, Tong Zhao, Jiaxin Xu, Tengfei Luo, and Meng Jiang. 2022. Graph Rationalization with Environment-based Augmentations. In *KDD '22: The 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Washington, DC, USA, August 14 - 18, 2022*, Aidong Zhang and Huzefa Rangwala (Eds.). ACM, 1069–1078. doi:10.1145/3534678.3539347

[38] Zhining Liu, Rana Ali Amjad, Ravinarayana Adkathimar, Tianxin Wei, and Hanghang Tong. 2025. SelfElicit: Your Language Model Secretly Knows Where is the Relevant Evidence. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2025, Vienna, Austria, July 27 - August 1, 2025*, Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (Eds.). Association for Computational Linguistics, 9153–9173. https://aclanthology.org/2025.acl-long.448/

[39] Zhining Liu, Ruizhong Qiu, Zhichen Zeng, Hyunsik Yoo, David Zhou, Zhe Xu, Yada Zhu, Kommy Weldemariam, Jingrui He, and Hanghang Tong. 2024. Class-Imbalanced Graph Learning without Class Rebalancing. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net. https://openreview.net/forum?id=pPnkpvBeZN

[40] Grégoire Mialon, Dexiong Chen, Margot Selosse, and Julien Mairal. 2021. GraphiT: Encoding Graph Structure in Transformers. *CoRR* abs/2106.05667 (2021). arXiv:2106.05667

[41] Erxue Min, Runfa Chen, Yatao Bian, Tingyang Xu, Kangfei Zhao, Wenbing Huang, Peilin Zhao, Junzhou Huang, Sophia Ananiadou, and Yu Rong. 2022. Transformer for Graphs: An Overview from Architecture Perspective. *CoRR* abs/2202.08455 (2022). arXiv:2202.08455

[42] Xuying Ning, Dongqi Fu, Tianxin Wei, Wujiang Xu, and Jingrui He. [n. d.]. Graph4MM: Weaving Multimodal Learning with Structural Information. In *Forty-second International Conference on Machine Learning*.

[43] Ruizhong Qiu, Dingsu Wang, Lei Ying, H. Vincent Poor, Yifang Zhang, and Hanghang Tong. 2023. Reconstructing Graph Diffusion History from a Single Snapshot. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, KDD 2023, Long Beach, CA, USA, August 6-10, 2023*, Ambuj K. Singh, Yizhou Sun, Leman Akoglu, Dimitrios Gunopulos, Xifeng Yan, Ravi Kumar, Fatma Ozcan, and Jieping Ye (Eds.). ACM, 1978–1988. doi:10.1145/3580305.3599488

[44] Ruizhong Qiu, Zhe Xu, Wenxuan Bao, and Hanghang Tong. 2025. Ask, and it shall be given: On the Turing completeness of prompting. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net. https://openreview.net/forum?id=AS8SPTyBgw

[45] Ruizhong Qiu, Weiliang Will Zeng, James Ezick, Christopher Lott, and Hanghang Tong. 2025. How efficient is LLM-generated code? A rigorous & high-standard benchmark. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net. https://openreview.net/forum?id=suz4utPr9Y

[46] Ladislav Rampásek, Michael Galkin, Vijay Prakash Dwivedi, Anh Tuan Luu, Guy Wolf, and Dominique Beaini. 2022. Recipe for a General, Powerful, Scalable Graph Transformer. In *NeurIPS*.

[47] Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.* 15, 1 (2014), 1929–1958. doi:10.5555/2627435.2670313

[48] Yongduo Sui, Caizhi Tang, Zhixuan Chu, Junfeng Fang, Yuan Gao, Qing Cui, Longfei Li, Jun Zhou, and Xiang Wang. 2024. Invariant Graph Learning for Causal Effect Estimation. In *Proceedings of the ACM on Web Conference 2024, WWW 2024, Singapore, May 13-17, 2024*, Tat-Seng Chua, Chong-Wah Ngo, Ravi Kumar, Hady W. Lauw, and Roy Ka-Wei Lee (Eds.). ACM, 2552–2562. doi:10.1145/3589334.3645549

[49] Yongduo Sui, Qitian Wu, Jiancan Wu, Qing Cui, Longfei Li, Jun Zhou, Xiang Wang, and Xiangnan He. 2023. Unleashing the Power of Graph Data Augmentation on Covariate Distribution Shift. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine (Eds.). http://papers.nips.cc/paper_files/paper/2023/hash/3a33ddacb2798fc7d83b8334d552e05a-Abstract-Conference.html

[50] Katherine Tieu, Dongqi Fu, Yada Zhu, Hendrik F. Hamann, and Jingrui He. 2024. Temporal Graph Neural Tangent Kernel with Graphon-Guaranteed. In *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, Amir Globersons, Lester Mackey, Danielle Belgrave, Angela Fan, Ulrich Paquet, Jakub M. Tomczak, and Cheng Zhang (Eds.). http://papers.nips.cc/paper_files/paper/2024/hash/abd3c6b90e474ec50a52c446926b00be-Abstract-Conference.html

[51] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett (Eds.). 5998–6008.

[52] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net.

[53] Dingsu Wang, Yuchen Yan, Ruizhong Qiu, Yada Zhu, Kaiyu Guan, Andrew Margenot, and Hanghang Tong. 2023. Networked Time Series Imputation via Position-aware Graph Enhanced Variational Autoencoders. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, KDD 2023, Long Beach, CA, USA, August 6-10, 2023*, Ambuj K. Singh, Yizhou Sun, Leman Akoglu, Dimitrios Gunopulos, Xifeng Yan, Ravi Kumar, Fatma Ozcan, and Jieping Ye (Eds.). ACM, 2256–2268. doi:10.1145/3580305.3599444

[54] Sinong Wang, Belinda Z. Li, Madian Khabsa, Han Fang, and Hao Ma. 2020. Linformer: Self-Attention with Linear Complexity. *CoRR* abs/2006.04768 (2020). arXiv:2006.04768

[55] Yingxin Wu, Xiang Wang, An Zhang, Xiangnan He, and Tat-Seng Chua. 2022. Discovering Invariant Rationales for Graph Neural Networks. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net.

[56] Zhanghao Wu, Paras Jain, Matthew A. Wright, Azalia Mirhoseini, Joseph E. Gonzalez, and Ion Stoica. 2022. Representing Long-Range Context for Graph Neural Networks with Global Attention. *CoRR* abs/2201.08821 (2022). arXiv:2201.08821

[57] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S. Yu. 2021. A Comprehensive Survey on Graph Neural Networks. *IEEE Trans. Neural Networks Learn. Syst.* 32, 1 (2021), 4–24. doi:10.1109/TNNLS.2020.2978386

[58] Haobo Xu, Yuchen Yan, Dingsu Wang, Zhe Xu, Zhichen Zeng, Tarek F. Abdelzaher, Jiawei Han, and Hanghang Tong. 2024. SLOG: An Inductive Spectral Graph Neural Network Beyond Polynomial Filter. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net. https://openreview.net/forum?id=0SrNCSklZx

[59] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2019. How Powerful are Graph Neural Networks?. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

[60] Peng Xu, Xiatian Zhu, and David A. Clifton. 2022. Multimodal Learning with Transformers: A Survey. *CoRR* abs/2206.06488 (2022). arXiv:2206.06488 doi:10.48550/arXiv.2206.06488

[61] Zhe Xu, Yuzhong Chen, Menghai Pan, Huiyuan Chen, Mahashweta Das, Hao Yang, and Hanghang Tong. 2023. Kernel Ridge Regression-Based Graph Dataset Distillation. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, KDD 2023, Long Beach, CA, USA, August 6-10, 2023*, Ambuj K. Singh, Yizhou Sun, Leman Akoglu, Dimitrios Gunopulos, Xifeng Yan, Ravi Kumar, Fatma Ozcan, and Jieping Ye (Eds.). ACM, 2850–2861. doi:10.1145/3580305.3599398

[62] Zhe Xu, Yuzhong Chen, Qinghai Zhou, Yuhang Wu, Menghai Pan, Hao Yang, and Hanghang Tong. 2023. Node Classification Beyond Homophily: Towards a General Solution. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, KDD 2023, Long Beach, CA, USA, August 6-10, 2023*, Ambuj K. Singh, Yizhou Sun, Leman Akoglu, Dimitrios Gunopulos, Xifeng Yan, Ravi Kumar, Fatma Ozcan, and Jieping Ye (Eds.). ACM, 2862–2873. doi:10.1145/3580305.3599446

[63] Zhe Xu, Kaize Ding, Yu-Xiong Wang, Huan Liu, and Hanghang Tong. 2022. Generalized Few-Shot Node Classification. In *IEEE International Conference on Data Mining, ICDM 2022, Orlando, FL, USA, November 28 - Dec. 1, 2022*, Xingquan Zhu, Sanjay Ranka, My T. Thai, Takashi Washio, and Xindong Wu (Eds.). IEEE, 608–617. doi:10.1109/ICDM54844.2022.00071

[64] Zhe Xu, Kaize Ding, Yu-Xiong Wang, Huan Liu, and Hanghang Tong. 2024. Generalized few-shot node classification: toward an uncertainty-based solution. *Knowl. Inf. Syst.* 66, 2 (2024), 1205–1229. doi:10.1007/S10115-023-01975-7

[65] Zhe Xu, Boxin Du, and Hanghang Tong. 2022. Graph Sanitation with Application to Node Classification. In *WWW '22: The ACM Web Conference 2022, Virtual Event, Lyon, France, April 25 - 29, 2022*, Frédérique Laforest, Raphaël Troncy, Elena Simperl, Deepak Agarwal, Aristides Gionis, Ivan Herman, and Lionel Médini (Eds.). ACM, 1136–1147. doi:10.1145/3485447.3512180

[66] Zhe Xu, Kaveh Hassani, Si Zhang, Hanqing Zeng, Michihiro Yasunaga, Limei Wang, Dongqi Fu, Ning Yao, Bo Long, and Hanghang Tong. 2024. How to make LLMs strong node classifiers? *arXiv preprint arXiv:2410.02296* (2024).

[67] Zhe Xu, Ruizhong Qiu, Yuzhong Chen, Huiyuan Chen, Xiran Fan, Menghai Pan, Zhichen Zeng, Mahashweta Das, and Hanghang Tong. 2024. Discrete-state Continuous-time Diffusion for Graph Generation. In *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, Amir Globersons, Lester Mackey, Danielle Belgrave, Angela Fan, Ulrich Paquet, Jakub M. Tomczak, and Cheng Zhang (Eds.). http://papers.nips.cc/paper_files/paper/2024/hash/

91813e5ddd9658b99be4c532e274b49c-Abstract-Conference.html

[68] Yuchen Yan, Yuzhong Chen, Huiyuan Chen, Xiaoting Li, Zhe Xu, Zhichen Zeng, Zhining Liu, and Hanghang Tong. 2024. THeGCN: Temporal Heterophilic Graph Convolutional Network. *CoRR* abs/2412.16435 (2024). arXiv:2412.16435 doi:10.48550/ARXIV.2412.16435

[69] Yuchen Yan, Yuzhong Chen, Huiyuan Chen, Minghua Xu, Mahashweta Das, Hao Yang, and Hanghang Tong. 2023. From Trainable Negative Depth to Edge Heterophily in Graphs. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine (Eds.). http://papers.nips.cc/paper_files/paper/2023/hash/de2d52c5cf2bea853ef39bb2e1535dde-Abstract-Conference.html

[70] Yuchen Yan, Yongyi Hu, Qinghai Zhou, Lihui Liu, Zhichen Zeng, Yuzhong Chen, Menghai Pan, Huiyuan Chen, Mahashweta Das, and Hanghang Tong. 2024. PaCEr: Network Embedding From Positional to Structural. In *Proceedings of the ACM on Web Conference 2024, WWW 2024, Singapore, May 13-17, 2024*, Tat-Seng Chua, Chong-Wah Ngo, Ravi Kumar, Hady W. Lauw, and Roy Ka-Wei Lee (Eds.). ACM, 2485–2496. doi:10.1145/3589334.3645516

[71] Yuchen Yan, Lihui Liu, Yikun Ban, Baoyu Jing, and Hanghang Tong. 2021. Dynamic Knowledge Graph Alignment. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*. AAAI Press, 4564–4572. doi:10.1609/AAAI.V35I5.16585

[72] Yuchen Yan, Si Zhang, and Hanghang Tong. 2021. BRIGHT: A Bridging Algorithm for Network Alignment. In *WWW '21: The Web Conference 2021, Virtual Event / Ljubljana, Slovenia, April 19-23, 2021*, Jure Leskovec, Marko Grobelnik, Marc Najork, Jie Tang, and Leila Zia (Eds.). ACM / IW3C2, 3907–3917. doi:10.1145/3442381.3450053

[73] Tianjun Yao, Yongqiang Chen, Zhenhao Chen, Kai Hu, Zhiqiang Shen, and Kun Zhang. 2024. Empowering Graph Invariance Learning with Deep Spurious Infomax. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net. https://openreview.net/forum?id=u9oSQtujCF

[74] Chengxuan Ying, Tianle Cai, Shengjie Luo, Shuxin Zheng, Guolin Ke, Di He, Yanming Shen, and Tie-Yan Liu. 2021. Do Transformers Really Perform Badly for Graph Representation?. In *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan (Eds.). 28877–28888.

[75] Hyunsik Yoo, Yeon-Chang Lee, Kijung Shin, and Sang-Wook Kim. 2022. Directed Network Embedding with Virtual Negative Edges. In *WSDM '22: The Fifteenth ACM International Conference on Web Search and Data Mining, Virtual Event / Tempe, AZ, USA, February 21 - 25, 2022*, K. Selcuk Candan, Huan Liu, Leman Akoglu, Xin Luna Dong, and Jiliang Tang (Eds.). ACM, 1291–1299. doi:10.1145/3488560.3498470

[76] Qi Yu, Zhichen Zeng, Yuchen Yan, Zhining Liu, Baoyu Jing, Ruizhong Qiu, Ariful Azad, and Hanghang Tong. 2025. PLANETALIGN: A Comprehensive Python Library for Benchmarking Network Alignment. *CoRR* abs/2505.21366 (2025). arXiv:2505.21366 doi:10.48550/ARXIV.2505.21366

[77] Linan Yue, Qi Liu, Ye Liu, Weibo Gao, Fangzhou Yao, and Wenfeng Li. 2024. Cooperative Classification and Rationalization for Graph Generalization. In *Proceedings of the ACM on Web Conference 2024, WWW 2024, Singapore, May 13-17, 2024*, Tat-Seng Chua, Chong-Wah Ngo, Ravi Kumar, Hady W. Lauw, and Roy Ka-Wei Lee (Eds.). ACM, 344–352. doi:10.1145/3589334.3645332

[78] Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontañón, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, and Amr Ahmed. 2020. Big Bird: Transformers for Longer Sequences. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin (Eds.).

[79] Zhichen Zeng, Boxin Du, Si Zhang, Yinglong Xia, Zhining Liu, and Hanghang Tong. 2024. Hierarchical Multi-Marginal Optimal Transport for Network Alignment. In *Thirty-Eighth AAAI Conference on Artificial Intelligence, AAAI 2024, Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence, IAAI 2024, Fourteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2014, February 20-27, 2024, Vancouver, Canada*, Michael J. Wooldridge, Jennifer G. Dy, and Sriraam Natarajan (Eds.). AAAI Press, 16660–16668. doi:10.1609/AAAI.V38I15.29605

[80] Zhichen Zeng, Ruizhong Qiu, Wenxuan Bao, Tianxin Wei, Xiao Lin, Yuchen Yan, Tarek F. Abdelzaher, Jiawei Han, and Hanghang Tong. 2025. Pave Your Own Path: Graph Gradual Domain Adaptation on Fused Gromov-Wasserstein Geodesics. *CoRR* abs/2505.12709 (2025). arXiv:2505.12709 doi:10.48550/ARXIV.2505.12709

[81] Zhichen Zeng, Ruizhong Qiu, Zhe Xu, Zhining Liu, Yuchen Yan, Tianxin Wei, Lei Ying, Jingrui He, and Hanghang Tong. 2024. Graph Mixup on Approximate Gromov-Wasserstein Geodesics. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net. https://openreview.net/forum?id=PKdege0U6Z

[82] Zhichen Zeng, Si Zhang, Yinglong Xia, and Hanghang Tong. 2023. PARROT: Position-Aware Regularized Optimal Transport for Network Alignment. In *Proceedings of the ACM Web Conference 2023, WWW 2023, Austin, TX, USA, 30 April 2023 - 4 May 2023*, Ying Ding, Jie Tang, Juan F. Sequeda, Lora Aroyo, Carlos Castillo, and Geert-Jan Houben (Eds.). ACM, 372–382. doi:10.1145/3543507.3583357

[83] Zhichen Zeng, Ruike Zhu, Yinglong Xia, Hanqing Zeng, and Hanghang Tong. 2023. Generative Graph Dictionary Learning. In *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA (Proceedings of Machine Learning Research, Vol. 202)*, Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (Eds.). PMLR, 40749–40769. https://proceedings.mlr.press/v202/zeng23c.html

[84] Zhenning Zhang, Boxin Du, and Hanghang Tong. 2022. SuGeR: A Subgraph-based Graph Convolutional Network Method for Bundle Recommendation. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management, Atlanta, GA, USA, October 17-21, 2022*, Mohammad Al Hasan and Li Xiong (Eds.). ACM, 4712–4716. doi:10.1145/3511808.3557707

[85] Lecheng Zheng, Dongqi Fu, Ross Maciejewski, and Jingrui He. 2024. DrGNN: Deep Residual Graph Neural Network with Contrastive Learning. *Trans. Mach. Learn. Res.* 2024 (2024). https://openreview.net/forum?id=frb6sLbACS

[86] Lecheng Zheng, Baoyu Jing, Zihao Li, Zhichen Zeng, Tianxin Wei, Mengting Ai, Xinrui He, Lihui Liu, Dongqi Fu, Jiaxuan You, Hanghang Tong, and Jingrui He. 2024. PyG-SSL: A Graph Self-Supervised Learning Toolkit. *CoRR* abs/2412.21151 (2024). arXiv:2412.21151 doi:10.48550/ARXIV.2412.21151

[87] Qinghai Zhou, Yuzhong Chen, Zhe Xu, Yuhang Wu, Menghai Pan, Mahashweta Das, Hao Yang, and Hanghang Tong. 2024. Graph Anomaly Detection with Adaptive Node Mixup. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management, CIKM 2024, Boise, ID, USA, October 21-25, 2024*, Edoardo Serra and Francesca Spezzano (Eds.). ACM, 3494–3504. doi:10.1145/3627673.3679577